Learning Location Embeddings from Species Distribution Models

Harry Lennox

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2022

Abstract

The feature vectors produced by deep neural networks can provide useful representations which can subsequently be fine-tuned to a related task. The aim of this project is to use the features of a network trained for Species Distribution Modelling and apply them to several other related tasks. Intuitively, we know that the presence or absence of a species within a specific location is dependent upon environmental properties such as tree cover or altitude, therefore these pre-trained features could help to predict such properties. We explore this task using both synthetic and real-world data and investigate the usefulness of input encodings for the Species Distribution Model in order to better represent points on a globe instead of a flat map.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Harry Lennox*)

Acknowledgements

First and foremost, I would like to thank my supervisor, Oisin Mac Aodha, for his vital assistance and willingness to help throughout the year. His guidance and support has been an enormous help to me, and I have learned a great deal from it.

Additionally, I am deeply grateful to all my friends and family for always being there to support me.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

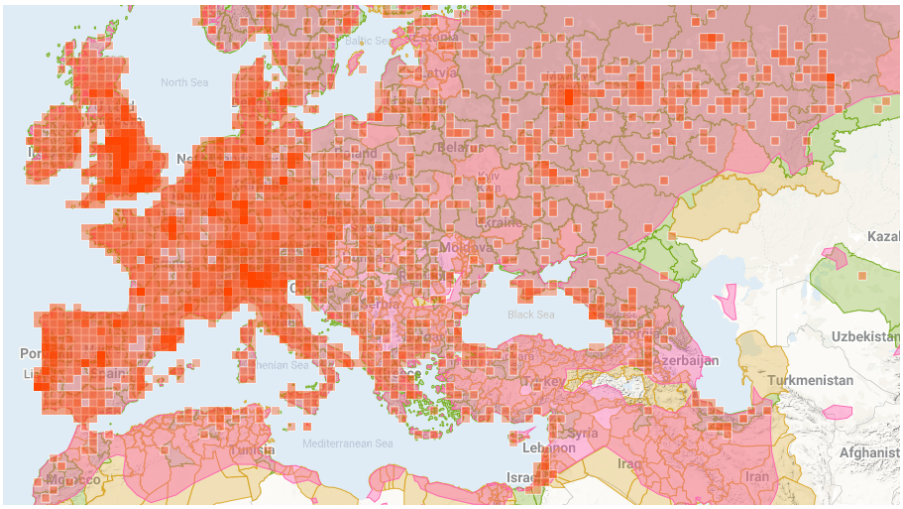


Figure 1.1: An example range map for the European Robin (Erithacus rubecula) taken from the iNaturalist website [1]. Areas of higher sightings are shown in darker red. The pink region shows a prediction of presence in unknown areas.

The study of ecology is an important area of research that helps us to understand the various ecosystems that we inhabit, and how they are changing due to human impact and climate change. Species Distribution Modelling (SDM) is an area of ecology focusing on recording the presence and absence of species across the planet, with these sightings often represented as a range map such as in the figure above. Through these spatio-temporal recordings, we can capture information on the regions a particular species inhabits, track their migration patterns throughout the year, monitor the changes in habitation over time and examine the relative abundance of species.

It has many real-life applications such as preservation of rare species and measuring human impact on environmental change. However, the task of gathering this data is a lengthy one that requires professionals in ecology to spend a large amount of time

surveying an area to establish the presence or absence of a species. This task is even more difficult for species that are difficult to spot, record, track and identify, such as various species of insect. These species may be more difficult or impossible to observe in some seasons (e.g. hibernation) or may lack a community that actively tries to observe them. Therefore, ecological research proceeds slowly and we often lack a ground truth value for large areas of rarely-explored territory. Until a species can be confirmed as absent from a location by specialists, there is a high uncertainty over its presence. To address such issues, various machine learning-based approaches have been proposed to accelerate the process of environmental research and SDM.

Neural network-based approaches to SDM result in a set of feature vectors, which can be seen as the model's learned representation of how a species is distributed across the map. In this paper, we will apply some intuition at this stage. In the real world, the places a species inhabits are not random, but are instead selected based on various environmental properties that the species prefers. Therefore, knowing which species inhabit an area could tell us a lot about the environmental makeup of that area. We propose that features learned through neural SDM can be very useful in modelling these different but related tasks. By pre-training on a SDM network, we can then fine-tune the produced features to a number of relevant environmental classification tasks such as altitude, tree cover, or population density estimation.

1.2 Contribution

This paper contributes a novel use of features produced from neural networks tasked with SDM, and aims to answer the following questions:

- Can representations learned from joint species distribution models be used for other environmental tasks via Transfer Learning?
- What is the impact of how we encode the input to these models on quality of the features learned when evaluated on the transfer tasks?

What we mean by 'encode' here is applying a transformation to a raw input that improves its representational power.

The traditional machine learning task we will be exploring is predicting the relationship between various environmental properties and the likelihood that a certain species is present there.

We show that pre-training on a SDM network can improve performance over a wide range of environmental tasks, and that both the number of species involved in pre-training and the way in which we encode inputs to such a network is vital.

1.3 Project Outline

We will begin by outlining the relevant research in the area of Species Distribution Modelling, as well as a few relevant techniques that we will be employing in our experiments. Next, we will explore the datasets used in the project: synthetic data,

iNaturalist 2021 and the Google Earth Engine. In Section 4 we will describe our model and the process in which we will train our species distribution model and fine-tune it to the various related tasks. Section 5 will cover our experiments using this model, and we will go into detail evaluating the effectiveness of different input mappings and comparing results from both the synthetic and iNaturalist datasets. We will discuss our results and potential reasons for notably good or poor outcomes. Finally, we will provide conclusions to our work, including areas for future improvement and research.

Chapter 2

Background

2.1 Deep Learning for SDM

SDM has been an area of research interest for many decades, but it is only more recently that machine learning approaches have been utilised to improve and automate the process of SDM. There are many properties to consider when producing such a model. Some model the distribution for a single species at a time, whilst other approaches consider many species at once and create a multi-species model, an area that has seen growth in recent years [27, 10]. Some recent approaches consider a joint distribution between all species, taking advantage of species co-occurrence to achieve more accurate results [10]. Additionally, there has been much work performed in understanding sources of error and uncertainty in these models, which can be sourced from both the observational data quality itself and the approximate nature of the model itself [22]. One of the key defining traits for a SDM model however is the data it relies upon - whether it is presence-absence or presence-only.

2.1.1 Presence/Absence Methods

Presence-absence methods have the advantage of knowledge of where a given species is present as well as where it is absent, but will always be bound by the speed at which the data can be collected across various remote areas of the world, where it is far more difficult to produce results. However, this data still exists in sufficient amounts for most areas to allow a wide range of machine learning processes to produce effective results. Many methods have been used, such as Gaussian processes [16], logistic regression techniques [15] and neural networks [6, 29]. Such models can still prove to be invaluable for researchers, providing more precise results and potentially pointing field workers towards highly probable areas that lack a presence or absence record.

2.1.2 Presence-only Methods

On the other end of the spectrum, several deep learning methods have been proposed to utilise the large and growing pool of community-driven presence-only resources available online, such as iNaturalist and eBird. Whilst this data is much easier to collect

in bulk, it does not contain any ground truth since there are no recorded absences. This lack of a ground truth has inspired various approaches from researchers, such as the usage of pseudo-absences [18, 8] though knowing how many absences to generate and where is an issue of its own [5].

The most relevant to our uses is the practice of embedding additional information into the model to record the richer details of relationships between species and the world. Intuitively, we know that the location a specific species can be found in is based upon several environmental factors such as climate, tree cover and other species (predator and prey relationships). There have been successful approaches focused on utilising such embeddings [19, 10, 27], producing features better able to capture the true distribution. However, there has been little work done in analysing the usefulness of such features for other related tasks.

2.1.3 Spatio-Temporal Distribution Modelling

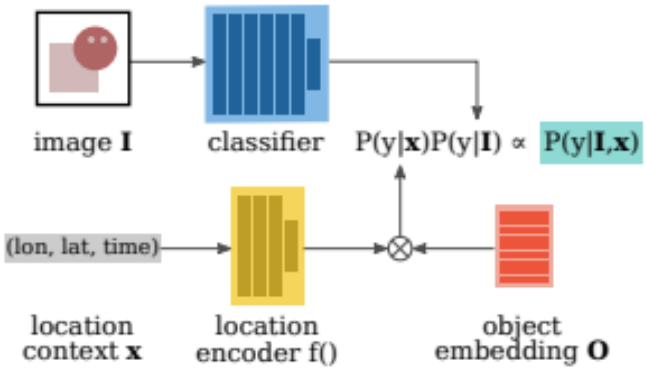


Figure 2.1: A figure explaining the structure of the network used by Mac Aodha et al. [19]

The basis for our analysis involves working on a spatio-temporal model of species distribution. In particular we will take inspiration from the spatio-temporal prior proposed by Mac Aodha et al. The model combines image classification with location embeddings to improve test-time classification performance, utilising our intuition that two species occurring in the same spatio-temporal area are unlikely to be independent. It achieves this through a spatio-temporal prior for representing the relationships between a location x and an object category y. If category y has been found to be likely at location x, the embedding will return a higher value in range 0 to 1, which is ultimately combined with the probability from image classification. Inspired by [7], we assume that the image I and its associated location x are conditionally independent from the species label y, therefore:

$$P(y|I,x) = \frac{P(I,x|y)P(y)}{P(I,x)} = \frac{P(I)P(x)}{P(I,x)} \frac{P(y|I)P(y|x)}{P(y)} \propto P(y|I)P(y|x) \tag{2.1}$$

This allows the image classifier $P(y|I)$ and spatio-temporal prior $P(y|x)$ to be represented separately.

This location embedding technique for a spatio-temporal prior was inspired by [10], where the idea of Deep Multi-Species Embedding was proposed to discover inner relationships between various species, modelling a joint distribution. Furthermore, more relevant to our purposes, species that co-occur across the world provide an implication that conditions in all such areas are similar. This relationship can provide a basis for predicting other values based on nothing more than what we know about the different species that live in a particular area. By leveraging the rich information learned by a neural network model, we can then utilise it for predicting other related metrics.

2.2 Transfer Learning

The practice of using a model's knowledge and applying it to a related problem is known as Transfer Learning [28], and is well-established and explored over a wide range of areas [13, 17, 30, 20], some of which we will review in the following section. It is a highly useful practice for problems where a large pre-training dataset exists that can then be adapted to a more precise task on a smaller dataset, and there is an intuitive natural principle behind it. It is much easier for the human mind to learn something new if it is related to something it already knows, for example learning a new language. A native English speaker will find it much easier to learn German compared to a language like Chinese.

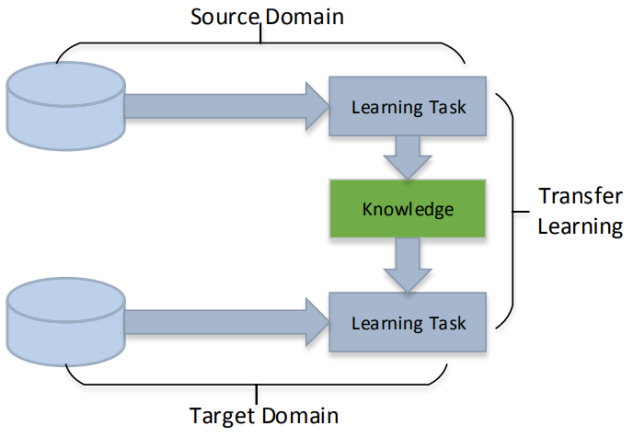


Figure 2.2: A visual representation of transfer learning [25].

2.2.1 Terminology

A domain D is made up of a feature space X and its corresponding marginal probability function $P(X)$. A task T is made up of labels Y and a function f for predicting the label associated with a given x. For domains D_S and D_T with corresponding tasks T_S and T_T, Transfer Learning can be formally defined as learning the predictive function f_T utilising information learned from D_S and T_S.

2.2.2 Neural Network Application

This principle can then be applied to deep neural networks, taking a network trained on a large dataset and applying it to a related but different task. Transfer learning can be homogeneous, where the features and labels of the source and target domain are the same, or heterogeneous where the opposite is true. Several such networks exist that are pre-trained on expansive datasets for various tasks, such as ImageNet [11] for image data and GloVe for natural language [21] that some models use as a source domain for transfer learning due to their excellent performance over a wide range of classes.

When a large model is trained on a general dataset aimed to assist in various other tasks, this is known as pre-training. The Species Distribution Model we will train is one such model. When pre-trained features are used as a basis for a different task T_S, this is known as fine-tuning. We will fine-tune our SDM features to predict various other environmental properties.

2.2.3 Performance of Transfer Learning

A question that has received much attention in the area of transfer learning is how to achieve optimal performance. Does pre-training on more data improve accuracy? Is there evidence for more fine-grained source data performing better on a fine-grained target task? Yosinski et al. [31] showed that task similarity does have an effect on transfer learning, following our intuition. By considering subsets of ImageNet, split on whether they contain man-made or natural objects to be purposely dissimilar, performance was shown to degrade. Building on this, Ngiam et al. [20] provide some intuitive and helpful conclusions. The size of the source dataset was found to be less important, with accuracy improving when discounting irrelevant samples. In our setting, this could mean ignoring a single rogue sighting for a rare species far away from where it is primarily clustered. Additionally, and relevant to our purposes, transfer performance was found to depend upon the source network capturing similar discriminative factors to the target. This is particular to fine-grained situations such as Species Distribution Modelling.

In our scenario, there is a close relationship between the distribution of a species and the features of the world around the areas it inhabits. Therefore, by the above, we have both intuitive and evidence-based reasoning for the plausibility of transfer learning using species distribution models. However, areas with less data on environmental variables and species sightings could negatively affect our model performance, as we have less confidence in our predictions. This is an issue faced by all models utilising ecological research data, as there are some parts of the world far more inaccessible for research than others.

2.3 Satellite Imagery

Previous work has utilised the vast amount of detailed, satellite imagery data from across the world in order to identify geographical features. Rolf et al. [23] proposed a method for using such data to estimate various geographical and human-centric features,

many of which we are interested in being able to predict with our model. A downside of using satellite imagery is the size of data required, requiring greater computing resources and storage capacity. Such models provide a good benchmark to compare our results to, and coming close to the the level of accuracy produced here would be a strong argument for the usefulness of SDM features in other tasks. We will not, however, be utilising satellite images, nor any images in our model.

2.4 Multi-Task Learning

Multi-task learning (MTL) is an approach where domain information over multiple tasks is applied as an inductive bias, improving generalisation using a shared representation [24]. The intuition behind this method is that by exploiting commonalities between different tasks, similarly to transfer learning, we can use what is learned from one task to improve upon prediction accuracy for other tasks that we learn in parallel.

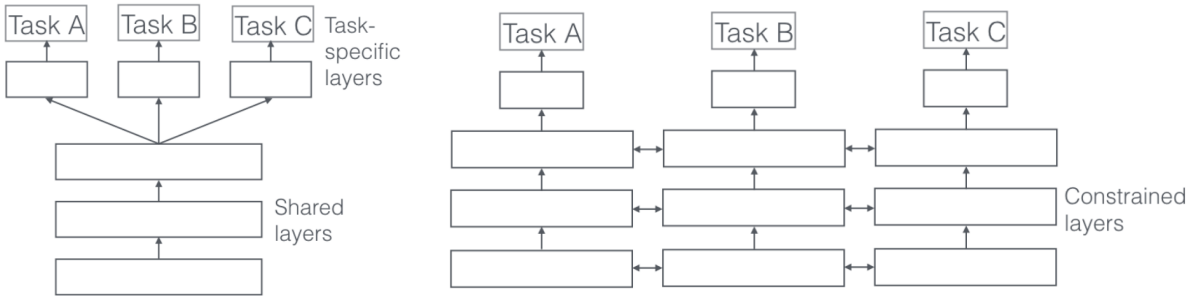


Figure 2.3: A graphical interpretation of hard (left) and soft (right) parameter sharing by Ruder et al. [24]

More concretely, we can apply MTL through either hard or soft parameter sharing. In hard parameter sharing, the hidden layers of the network are shared between all tasks and separated into multiple output layers specific to their task. In soft parameter sharing, there is a model with parameters for each task and these parameters are regularised to encourage similarity between the models.

In our situation, we have already established that there is a link between the environmental parameters that we wish to predict. For example, in general it would be expected that increased population density results in less tree coverage and greater likelihood of a lower altitude. By learning these tasks in parallel, we could utilise such properties, however this is an experiment we will leave to future research, instead learning each task individually.

Chapter 3

Datasets

3.1 Synthetic Data

3.1.1 Usage

Initial experiments on the neural network were performed on synthetic data - a dataset procedurally generated by code. The details of how this data can be generated are below, but the reasoning for using synthetic data is also important. Firstly, it was more lightweight for initial bug-fixing and such when setting up the network. There was no need to read from massive files, and this simplified the task at a stage where understanding the problem was key. Additionally, it was simple to write code in such a way that, in theory, the new dataset could simply be 'plugged in' and it would work just the same. Of course, in reality there were difficulties with this, covered in the next section. The synthetic data case is an easier setting for the network, since there are many oddities in real-world data that could confuse it. This makes it useful as a frame of reference for further experiments and as a proof of concept in the early stages of the project. Most importantly, when working with synthetic data we have access to something we don't have with real data - the objective ground truth, both for where a species is and is not present. This allows us to use metrics and perform other analysis on the synthetic data that we could not perform on real data.

3.1.2 Parameters

The synthetic data creation procedure was created to be a simple representation of species distributions, whilst also capturing as many traits of the real world data as possible. It can be split into two parts - generating presences and generating pseudo-absences. The size of the 'map' is intended to represent real world (lat, long) values, so it ranges from -180 to 180 along the x-axis and -90 to 90 along the y-axis. There are three main parameters for generating the data - NUM_CLASS, how many classes (species) to generate, NUM_DATA, the max number of data points a single class may have and NUM_ABS, the number of absences to generate per species. All other parameters are randomly generated according to the procedure below:

3.1.3 Generating Presences

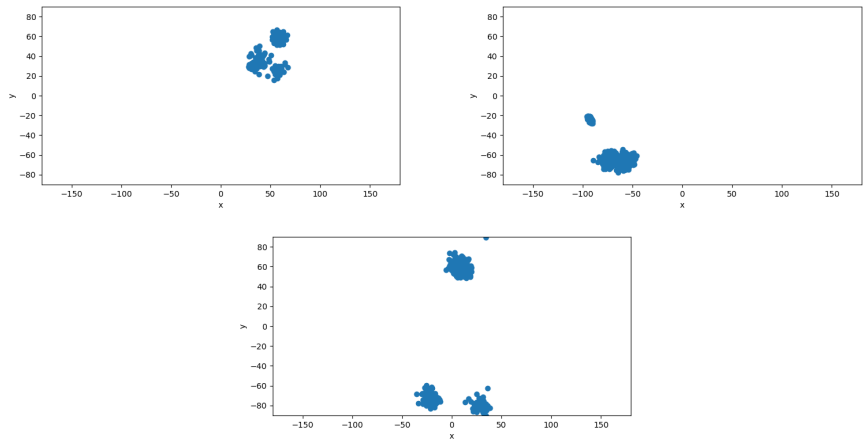


Figure 3.1: Some examples of our synthetic data, generated with NUM_CLASS=100 and NUM_DATA = 240.

To generate the presences, our sightings of species, we use a Gaussian model. Firstly, five 'centre' points or 'clusters' are initialised across the map. These can be thought of as distinct and important natural areas, such as the Amazon rainforest where thousands of unique species reside. For each species we wish to generate, follow the procedure below:

- Randomly assign it a cluster to belong to.
- Randomly assign it a number of Gaussians between 1 and 4. Note: This is not a uniform choice, and probability decreases for larger numbers of Gaussians.
- Randomly assign it the 'migratory' trait, with a low probability.
- Randomly assign each Gaussian a mean, covariance and number of data points to generate between 1 and NUM_DATA. Note, the mean will be the centre of a cluster, offset by a randomly generated value for x and y. The exception to this is if the 'migratory' trait is true, in which case the species is not bound to a cluster and its Gaussian's means can be anywhere on the map.
- Generate the specified number of data points according to the Gaussian parameters, and assign these to an array.

This gives us a list of species, each with their own distributions, which can then be used in our neural network model.

3.1.4 Generating Equal Presences

We may also wish to experiment on a dataset where every class has an equal amount of data, and so we have a similar method for producing such a dataset. This works identically to the above process, except we remove the constraint of having to belong to a cluster, essentially giving every species the migratory trait and allowing for Gaussians

anywhere on the map. We also generate exactly 50 samples per class, to mimic a version of the iNaturalist dataset we will explore in the next section.

3.1.5 Generating Pseudo-Absences

Generating the pseudo-absences is a much simpler process, since these are completely random points on the map not constrained by any properties. For every species we wish to generate, absences are generated as shown below:

- Repeat the following NUM_ABS times...
- Randomly choose (x, y) values from the whole range of the map (between -180 to 180 and -90 to 90).
- Assign the point to an array, as well as its corresponding class label.

The absence values can be indexed by multiplying the species label by NUM_ABS, which will locate the first absence for that class. A number between 1 and NUM_ABS can then be added to index over the other absences for this class. An example of this data can be seen in Figure 3.2. The uses for these pseudo-absences will be explained in the next chapter when we cover the neural network architecture.

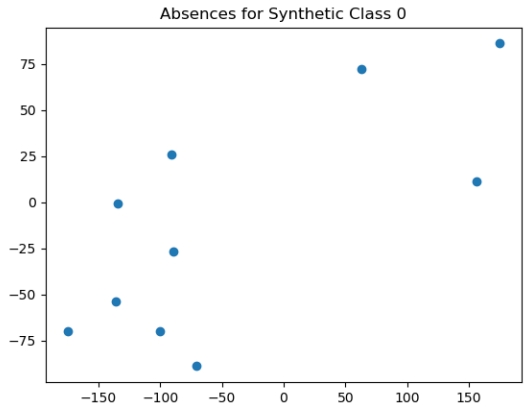


Figure 3.2: Absences for a synthetic class - for this example NUM_ABS=10.

3.2 The iNaturalist Dataset

3.2.1 Description of the dataset

iNaturalist [4] is a community-driven website for recording images of various species from millions of users across the world, and has been active since 2008. The images are accompanied by metadata recording the exact location and time of the sighting, and split into different species categories and supercategories. When collated into this large dataset, its community-driven nature provides plentiful information for neural network-based approaches. The 2021 version of the dataset used in this project contains 2.7 million training samples, but also offers a 'mini' training set that includes 500k samples

- 50 from the 10000 different species contained in the full dataset. In addition, 100k validation images and 500k test images are also provided, along with the corresponding positional data. Appendix A provides a table containing the full details, taken from the iNat 2021 competition. This data is presence-only, and so models trained on iNat will have no knowledge of ground truth absence. The speed at which new data can be acquired is far faster than relying on professional recordings due to the community-driven nature and accessibility, but this dataset does not come without flaws.

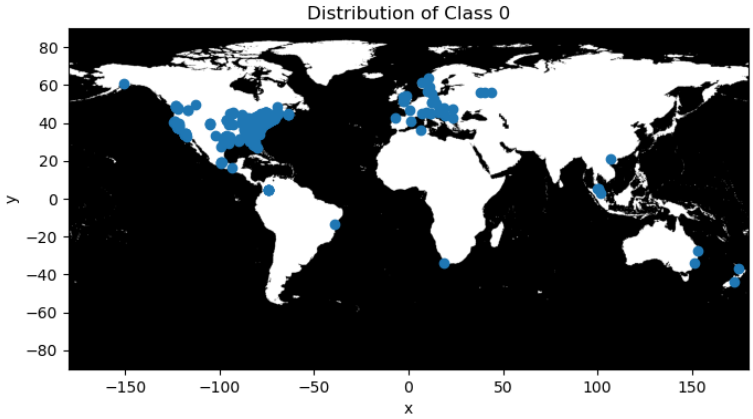


Figure 3.3: Example of a class taken from the iNaturalist dataset. This is class 0, Common Earthworm, though it seems to only be 'common' in Europe and North America.

3.2.2 Suitability and Limitations

The 'Plants' supercategory contains hundreds of thousands of entries, however the 'Arachnids' supercategory contains just over 40,000. The primary reason for this is that it is simply more difficult to record pictures of some naturally occurring species than others, which can be due to factors such as their rarity, size, or the location they naturally inhabit. A community of ordinary people are far more likely to record what appears around their local environment and are actively encouraged to do so. iNaturalist hold a yearly City Nature Challenge, encouraging cities around the world to compete in submitting sightings of wildlife around their city. This provides a heavy bias towards urban environments, as well as a large temporal bias. Users are more likely to upload a sighting around the timing of the competition. Even without this factor however, there is an additional bias of people being less likely to go outside during winter months as opposed to summer. It has also been shown that the top 1% of users provide 62% of all observations [12], which will also affect the distribution of data towards the places these users are commonly recording observations. This is not a bias limited to this dataset, but instead a feature of citizen science datasets in general. Whilst it is important to be aware of the flaws, such data is still extremely valuable for researchers.

3.2.3 Citizen Science Dataset Research

A potential solution to this issue was proposed by [14] where they combine presence-only data with complimentary presence-absence data for the same species, maximising

a joint likelihood to correct for the sampling bias in the presence-only data. The intuition is that by sharing data, more reliable estimates of the species distribution can be obtained. Whilst it was shown to be an effective solution, it is still reliant upon the presence-absence data created by ecological surveys which may not be plentiful for every species on the planet. Since our model is trained on presence-only data alone, this method would not be practical for our purposes.

A more recent potential solution to the bias of citizen science datasets is explored by [9], which proposes End-to-End Shift Learning to reduce bias by attempting to learn and correct for the shift from scientific objective to biased data through a Shift Compensation Network. This is an attractive solution, as it was shown to outperform all other bias correction models - it would be an interesting focus for future work to investigate the usefulness of such a method on a species distribution model.

3.2.4 Technical Details and Import Process

The iNaturalist dataset comes in JSON format, with a lot of extra information that we are not interested in such as licenses, image size, year created, etc. It contains five different structures - info, images, categories, annotations and licenses. The latter four of these are lists, of objects for their corresponding class, whilst info describes the dataset as a whole. The full details of this dataset and its representation are included in the Github page for the dataset [3]. The first step in utilising this data is to pull out the (lat, long) values we need for training. These are contained within the 'images' list, and each entry also has an id which we will additionally be extracting for later use. This can then be used as our training data, but we still require the corresponding labels. These are contained within annotations, which consists of id, image_id and category_id. Id and image_id have been observed to be the same value, and match the corresponding image using the id value we extracted earlier. The category_id is the label value we are interested in.

This is where we encounter an issue with the dataset - it contains missing entries where the (lat, long) values for an images are 'None'. This would immediately cause an error if used in training, therefore we must filter out these corrupt values. Once this was completed, it was found there were 1571 total corrupt values in the train_mini dataset alone. The corresponding labels were also removed, using the id of the corrupt images to identify them. The resulting 498,429 training samples (in the case of the 'mini' dataset) and labels are then saved to a file to save time in later runs. This same procedure is performed on the iNat validation data, and the same issues persist here, as well as with the full dataset where there are even more missing values.

3.3 Transfer Learning Data

For transfer learning, we decided upon the following 10 prediction tasks for environmental properties across the globe:

- Above Ground Carbon - Above-ground living biomass carbon stock density of combined woody and herbaceous cover.

- Elevation - Altitude from -457 to 8746 metres.
- Leaf Area Index - The leaf area in comparison to ground or trunk area of plants.
- Night Time Lights - The average of the visible band digital number values with no further filtering.
- Non-Tree Vegetation - Percentage of non-tree vegetation (e.g. grass, flowers) in an area.
- Non-Vegetated - Percentage of an area that is not vegetated.
- Population Density - Measures the estimated number of people per km^2
- Snow Cover - Percentage of an area that is covered by snow.
- Soil Moisture - Level of moisture in the soil across an area.
- Tree Cover - Percentage of an area that is covered by trees.

The data for these 10 tasks was extracted from the Google Earth Engine API [2] which provides a large collection of publicly available datasets for various properties - including many geophysical properties that we are interested in for the purposes of testing our SDM features. All of the above tasks relate to the species that might inhabit an area. Herbivores will likely live in an area with more vegetation, a snow-covered area will see more species with fur or thick fat, and species that live in areas of high population density are likely to be scavengers such as birds and rodents. We aim to test each of these tasks and take the mean precision achieved over all of them as the final metric for judging how well our model performs.

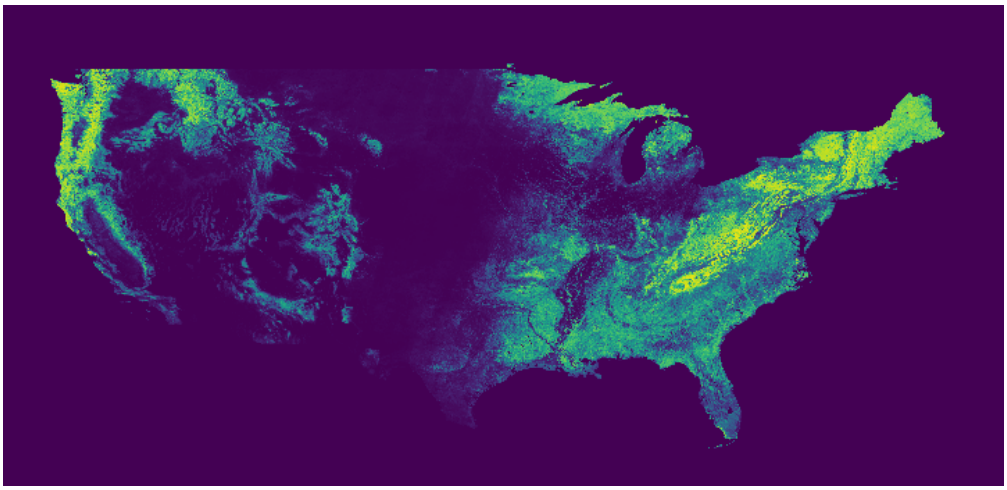


Figure 3.4: An example of the ground truth data for transfer learning tasks, masked to just the USA. This example shows the Tree Cover task. Colours closer to yellow represent higher values of tree cover, and descend in colour as tree cover decreases.

Chapter 4

Method

4.1 Problem Overview

As mentioned in the previous section, the traditional ML task we will explore is to predict the relationship between some environmental properties and the likelihood that a certain species is present there. This will be done in two parts - firstly we will require a model to predict the species likelihood from a dataset of (latitude, longitude) pairs covered in the previous section. Then, the features produced by this model will be fine-tuned to predict the various properties we are interested in.

Here we will introduce some basic notation that will remain standard throughout this section. First we introduce four hyperparameters. NUM_CLASS sets the number of classes (species) that will be used in classification, and NUM_ABS sets the number of pseudo-negatives to generate per class (this is only used initialisation). MAPPING_TYPE is an enum representing the type of input mapping to use, or None if we wish to use raw (lat, long) inputs. DATA_TYPE is another enum that represents which type of input data to use, it can be set to initialise or load existing data, either synthetic or iNat, mini or full, and also controls whether to output only features for transfer learning, or output probabilities for SDM prediction. This is used during transfer learning. For inputs, $\mathbf{x}_{\mathbf{train}}$ and $\mathbf{y}_{\mathbf{train}}$ represent the training data and labels respectively, and $\mathbf{x}_{\mathbf{val}}$ and $\mathbf{y}_{\mathbf{val}}$ represent the validation data and labels. Lastly, $\mathbf{x}_{\mathbf{absences}}$ and $\mathbf{y}_{\mathbf{absences}}$ represent pseudo-negative values for each class, a concept we will cover in the following sections.

When describing the architecture of a neural network, the following notation will be used.

- x_i - The input data, indexed by i. The input can be any length, depending on what encoding we use - unencoded would be a simple (lat, long) tuple.
- h_i^j - Hidden layer j, where i is the index of the specific hidden unit in layer i.
- f - Forward pass through the network.
- g_c - The output function - in our case this is the softmax function, where c indexes over the resulting vector.

- x_r - A randomly selected pseudo-negative for the relevant class from $\mathbf{x_{absences}}$. These will be encoded (lat, long) values used by the loss function.

Figure 4.1 below provides a graphical representation of the full process, from data initialisation to pre-training and finally fine-tuning and output predictions. Each step will be covered in greater detail in the sections following.

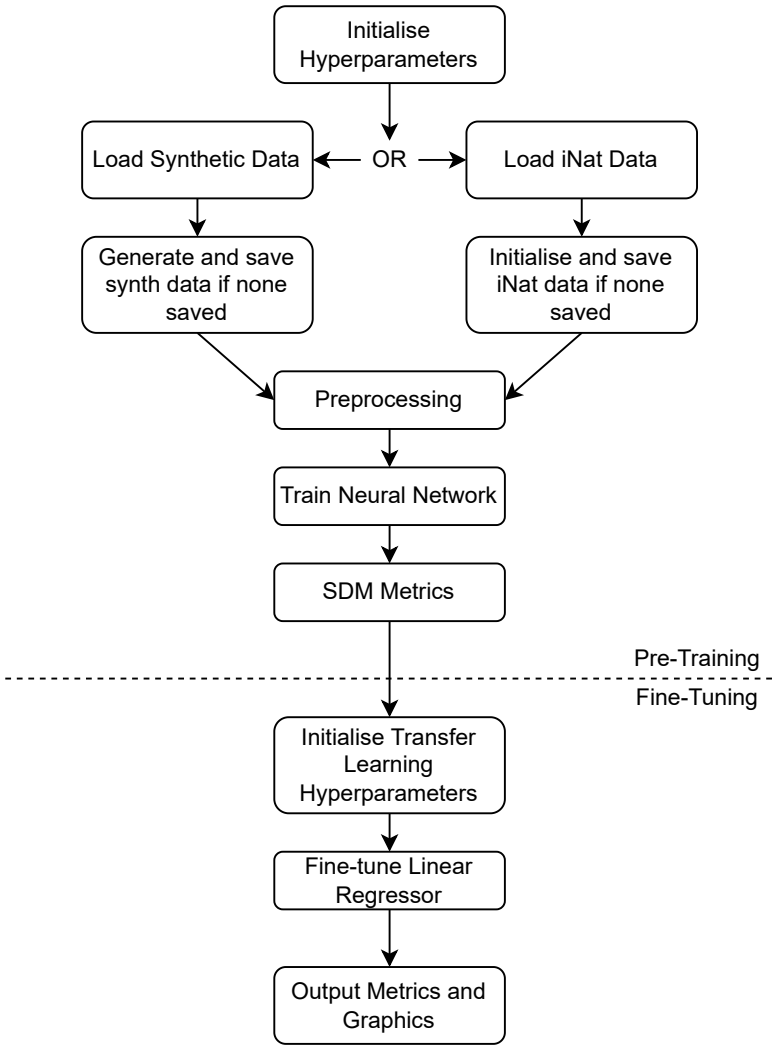


Figure 4.1: A representation of the full process

4.2 Neural Network Distribution Estimation

4.2.1 Neural Network Model

A relatively simple structure feed-forward structure mimicking the one used by Mac Aodha et al. [19] was decided upon for the neural network model to avoid complication and to allow for further experimentation in the future. If we later wish to know whether a more complex network with more inputs, or a completely different architecture will produce features better suited to other tasks, it will be useful to have baseline features such as these to compare to.

PyTorch was chosen as a library for the implementation of the model as it provides a widely-used, well-documented and powerful set of tools that also allows for networks to run on the GPU, massively speeding up running time.

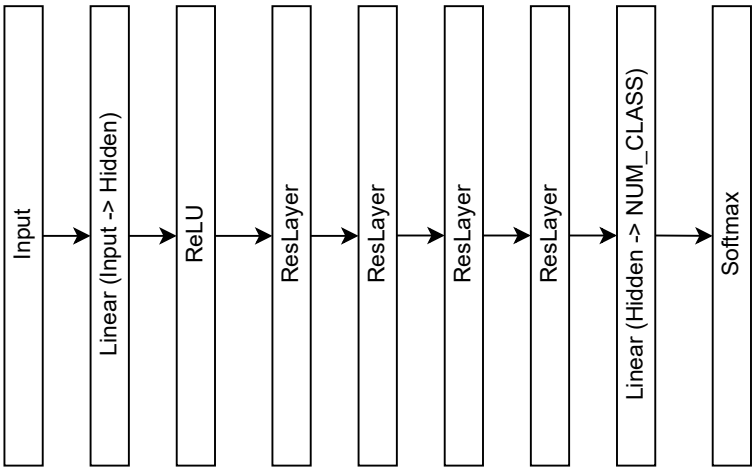


Figure 4.2: A diagram depicting the full NN structure.

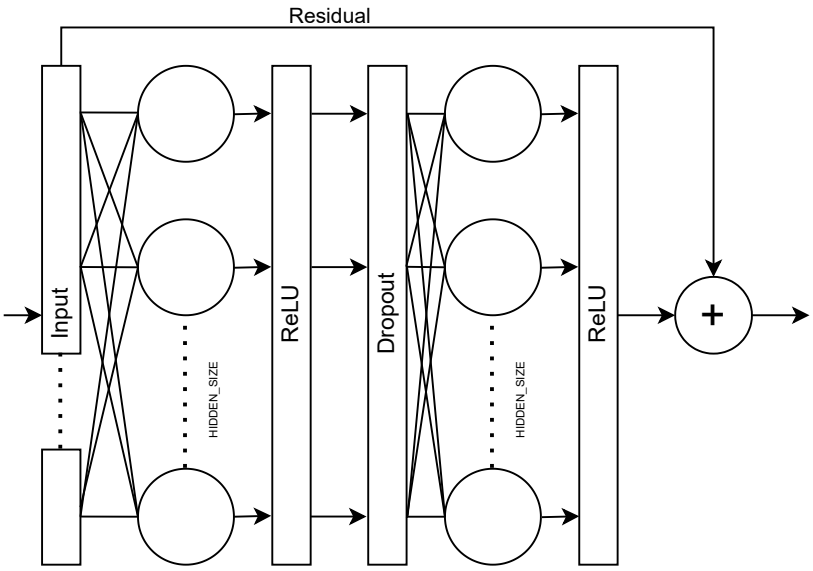


Figure 4.3: The structure of one ResLayer unit.

Our network is formed of one linear input layer, followed by a ReLU activation function and four sequential ResLayer blocks, and finally a linear output layer that projects to the number of classes we are experimenting on, before being sent through a softmax to generate output probabilities for each species. This final projection and softmax will not be performed if we want to output features, which will instead come directly from the final ResLayer. All linear layers are of the same hidden size. One ResLayer is comprised of a residual connection, two linear layers with ReLU activation functions, and a dropout layer between them. This gives us a total of 10 linear layers in the model as a whole, allowing the network to learn more complex functions whilst the dropout layers and residual connection work to reduce overfitting.

4.2.2 Presence-Absence Loss

A custom loss function was implemented for the network, based on the geo prior loss function presented by Mac Aodha et al [19]. This function assigns a negative loss, a positive loss and a background loss, which are simply added together to achieve the final loss. This loss function is shown below, where c is the true class of the data point x:

$$-[\sum_{C \neq c}(1 - g_C(f(x))) + \lambda log(g_c(f(x))) + log(1 - g_c(f(x)))] \tag{4.1}$$

The first part of this function, the negative loss, is a simple log loss applied at every index except that of the correct class. The closer its predictions for these classes are to 0, the lower the resulting loss. This encourages the network not to predict high values for every class, and instead to pick one it is confident in. The second part, the positive loss, performs the opposite task. It is applied to the index of the correct class, and the closer this value is to 1, the smaller the loss will be. This enables the network to learn the correct labels for a given input. Lambda is a hyperparameter that gives more importance to this positive loss, and similarly to [19] we set it to the number of classes in our dataset.

The final part is the background loss, where we will select from our list of absences a random point x_r for the class c. We utilise the assumption made by Mac Aodha et al [19]. Namely, that each class c exists within a relatively small subsection of the full space, therefore the chance of a randomly generated point being valid for a class is also small. Therefore, we can generate these pseudo-negatives for use in the loss function despite our input being presence-only data. We put this random point through the network, and assert that the loss for this point will be smaller the closer the network's output is to 0. This has the effect of focusing the network's attention on the relevant area of the map, so that it doesn't predict high values for every single point. These pseudo-negatives do the work of ground truth absences in our model, allowing us to use presence-only data.

A basic input to this loss function would be a (lat, long) pair, however an input x could also be any of the input mappings explained in the next section. So long as the input is some representation of our data, it does not matter how its dimensions or exact values have changed.

4.3 Preprocessing and Input Mapping

In this section we will discuss the preprocessing performed on the data before it is used within the neural network model, with a particular focus on input mappings - transformations we apply to the data in order to better represent the structure of a globe.

4.3.1 Why use input mapping?

When we look at a map, we are looking at a projection of the globe. Intuitively we know that where one side of the map ends, it wraps around to the other end. However,

there is no way for a neural network to know this - it simply takes in vectors. Therefore, there is a necessity to model the wrap-around nature of our data if we want to be able to recognise two points are close when they are near the edges of the map. For example, point X = (170, 20) and point Y = (-170, 20) would seem very far away on a basic grid without our globe assumption, however if we know that the grid wraps around at 180 degrees on the x-axis, then we can tell point X and Y are actually very close. Input mapping is a method to represent this relationship through transformations to the data and we will be using two varieties.

4.3.2 Basic Mapping

The basic form of input mapping used in our model consists of a sin-cos mapping, displayed below:

$$\gamma(x) = [cos(\pi x), sin(\pi x)] \tag{4.2}$$

This increases the number of dimensions of our input data from 2 to 4, since we concatenate the cos and sin values of the input. It also achieves the goal described above, since cosine has the property that cos(x) = cos(-x), as the function 'wraps around' in the same way the edges of our map do. Sine assists with this, as it has the property that -sin(x) = sin(-x), which allows us to still differentiate between a point on either side of the map. Together they give us a representation that should be similar for our points X and Y, but importantly not identical. For reference, passing those points through such a function would result in X = (-0.9946, 0.9984, 0.1033, 0.8896) and Y = (-0.9946, 0.9984, -0.1033, 0.8896). They are almost identical, showing their close proximity, but the third element is negative, so the network won't see them as the same point.

4.3.3 Fourier Mapping

A more advanced input mapping is the Fourier Mapping, which is similar in nature but involves an additional matrix B, which is initialised with values randomly sampled from a normal distribution centred at zero. In our implementation this is done with numpy.random.randn for an array of shape (SIZE, 2). The values are multiplied by SCALE to achieve the final B matrix. This has been shown to work well for coordinate-based neural networks, explained through the lens of neural tanget kernel theory by Tancik et al [26]. The variance of this distribution can vary, as well as the dimensions of the matrix, giving us two hyperparameters to adjust. The formula for this mapping is shown below:

$$\gamma(x) = [cos(\pi Bx), sin(\pi Bx)] \tag{4.3}$$

Now, the number of dimensions of our input data depends upon the size of B multiplied by 2 since we still concatenate as previously. The size hyperparameter controls the shape of the B matrix, and the higher this value is, the higher-dimensional the input data becomes. Increasing this parameter will generally give the network more information to work with, allowing for more precise prediction boundaries to be created.

The second hyperparameter is scale - this controls the scale of the normal distribution. More concretely, it is the standard deviation of the Gaussian curve from which values are drawn uniformly at random. The higher this value, the more uniform the input data will become. Essentially this controls the extent of the randomness at play. We will perform experiments on the Fourier encoding in the next chapter in order to determine the optimal values for both parameters.

4.3.4 Additional Preprocessing

Before passing the raw (lat, long) values through the input mapping, we first divide both values to normalise to the range(-1, 1). To do this we simply divide the latitude values by 180 and the longitude values by 90. Once the input mapping has been applied, we transform our input vectors to a tensor, later used to create the training and validation data-loaders.

4.4 Geographical Transfer Learning

4.4.1 Fine-tuning Process

The fine-tuning step takes various inputs in order to specify what tasks it will be performing over what area. Firstly, we may wish to use a mask - this allows us to focus on just one area of the globe, which reduces training time. Secondly, we provide a list of datasets (tasks) to train on - we will be training a separate model for each of these tasks. Lastly, we provide the model an algorithm for extracting features - this could be a basic (lat, long) representation, or a Fourier-encoded input mapping, or it could be features produced from our pre-trained SDM neural network.

Next, a simple linear regressor will be trained for each task and each feature extraction algorithm it has been provided and return a trained model for each. From these we can then output the prediction scores for the training and testing set over all tasks for each method. The final metric we will use to compare each model will be the mean score over all 10 prediction tasks. Lastly, we output an image showing the decision boundaries produced by the model, which we can then compare to the ground truth image.

4.4.2 Train-Test Split

Since there is only one Earth, we need to split its surface into a training and test set, which is not a trivial task. If we naively remove large portions of the map as test data, we may never even see a species contained within that area during training. We also run the risk of biasing our model towards the area of the map it is trained on, then achieving poor results when it is tested on an area it is completely unfamiliar with.

To do this, we split the map in a checkerboard pattern, where the size of each square can be varied as a hyperparameter. We evenly split the map between training and test sets, going from left to right and top to bottom, iterating between a training square and a testing square. A visual example of this method covering just the USA can be seen

in Figure 4.4. By doing this we give the model a more accurate representation of the entire map that is not biased towards one particular area (if we ignore the inherent bias in the dataset).

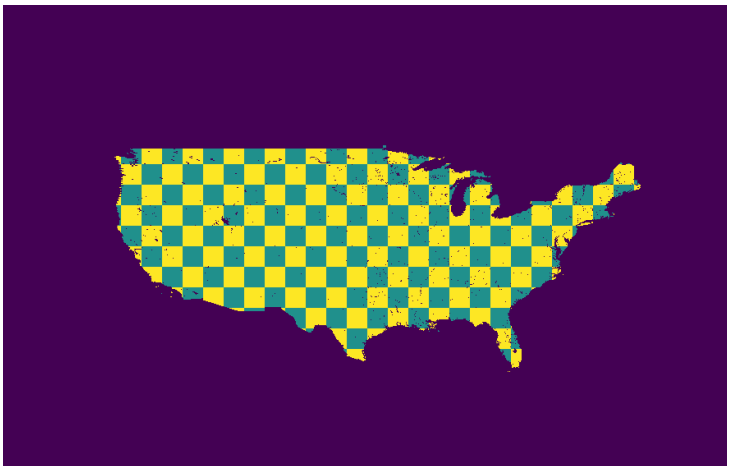


Figure 4.4: A visual representation of how we divide areas of the map into training and test sets. The blue squares are training areas, whilst the yellow squares are testing areas. The background dark colour is the mask, which covers all areas outside of the USA.

Chapter 5

Experiments and Discussion

5.1 Implementation Details

The following experiments were all performed on a network architecture as shown in Figure 4.1, with n=256 for all layers, NUM_ABS = 10 and NUM_CLASS varying for each experiment. A learning rate of 1e-3 was used with an Adam optimizer, and training was done over 16 epochs. Results for the species distribution modelling task can be measured in terms of average precision, the formula for which is shown below:

$$\sum_n (R_n - R_{n-1}) P_n \tag{5.1}$$

Where at the n'th threshold of the precision-recall curve, P_n and R_n are the precision and recall respectively. This function summarises the curve as a mean of precisions across the thresholds, weighted by recall. Since we do not have the ground truth data for iNat, we only use this metric for the synthetic data experiments. When we perform transfer learning, a different metric, R^2 is used:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{5.2}$$

$$RSS = \sum (y_{true} - y_{pred})^2 \tag{5.3}$$

$$TSS = \sum (y_{true} - \hat{y}_{true})^2 \tag{5.4}$$

RSS is the residual sum of squares, which measures the variance of error, and TSS is the total sum of squares, which measures the dispersion of points around the mean, where $\hat{y}$ indicates the mean of y_{true}. Together they form R^2, which is known as the coefficient of determination and measures how well a regression model fits the given data. The best possible score is a value of 1, and a model which always predicts the expected value of y would achieve a score of 0. Interestingly, this metric allows for negative scores, so performing exceptionally poorly on one task can greatly effect the mean over all tasks.

5.2 Fourier Encoding Optimisation

In order to perform experiments using the Fourier encoding, we must first find the ideal values for the size and scale parameters described in the Section 4.3.3. To do this, experiments were performed on the synthetic data with 100 classes. The low number of classes was to reduce computation time, and we choose synthetic data in order to get quantitative results. The goal of these experiments was to identify the trends as we increase both size and scale.

Size	Scale=1	Scale=2	Scale=4
16	0.546	0.5808	0.5668
32	0.5523	0.5749	0.5841
64	0.5489	0.5848	0.5896
128	0.5654	0.5904	0.5932
256	0.5722	0.5913	0.5942

Table 5.1: Experimental results for the different hyperparameter settings, measured in average precision.

The randomness inherent in the Fourier mapping causes variation in results that can interfere when spotting a trend. However, we can see that in general a higher size value leads to better results, though it has diminishing returns after a certain point, usually at around 128-256. This is likely due to the additional dimensions and randomness assisting the network in generating more complex decision boundaries, but eventually reaching a point where the additional information does little more to help. Therefore, we settled on 128 as the size value for future experiments, as we do not want our representation to become overly complex for little gain.

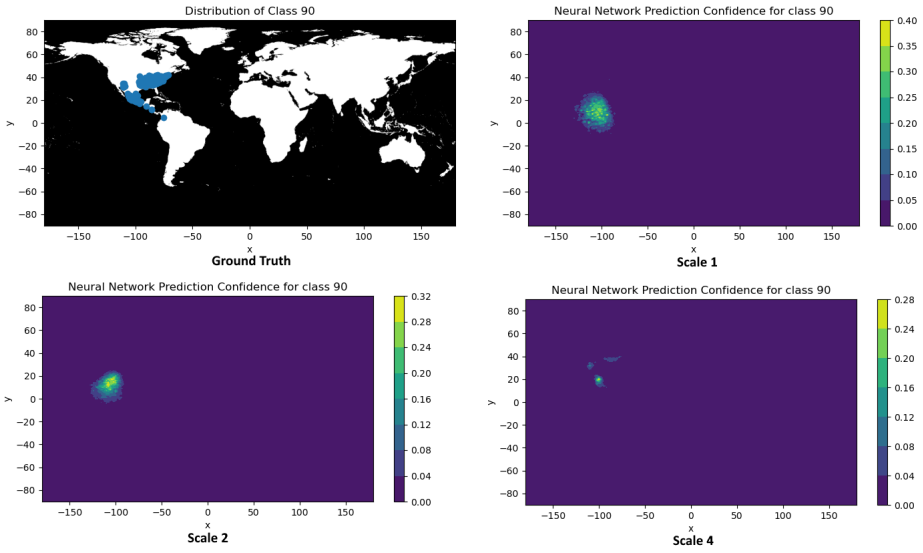


Figure 5.1: A comparison of decision boundaries for different scales. A static size of 32 was used for the experiments, with only scale changing.

For the scale hyperparameter, 4 was the most effective setting. If the B matrix is of lesser scale then the usefulness of the randomness is degraded, however the more we increase scale the greater the randomness, and increasing much higher could lead to the model not generalising well enough. This can also be seen in the generated decision boundaries in Figure 5.1 above, which becomes too precise and influenced by randomness at the point of scale 4. We wish to preserve a more balanced decision boundary with slightly less inherent randomness such as the ones displayed for scales 1 and 2, whilst maximising performance. Therefore, we will select scale 2 instead of scale 4. The results from these tests give us an optimal value for the Fourier mapping of size=128 and scale=2, and these values will be used in all future experiments for the Fourier encoding, unless stated otherwise.

5.3 Synthetic Data Evaluation

Dataset Size	No Encoding	Basic Encoding	Fourier Encoding
100	0.4136	0.4738	0.5804
500	0.159	0.1874	0.2307
1000	0.0998	0.1117	0.1399
2000	0.0592	0.0684	0.0832

Table 5.2: Average precision of the various input encodings for synthetic data.

Next, we perform experiments on our synthetic dataset and investigate the impact of number of classes and input encoding. For these experiments, and all following, the maximum number of species we will use is 2000 - this is simply since training can take an exceedingly long time past this point. The results shown in Table 5.2 paint a clear picture of the usefulness of input encodings for the species modelling task, especially for situations where we are only modelling relatively few species, where an input encoding can result in significant improvements to average precision.

As we can see in Figure 5.2 (next page), the neural network's confidence greatly increases after we apply the fourier encoding, but it grows less confident in the right half of the class's distribution. This is likely because there are other classes in that area that it is more confident in, and less on the left half, so it focuses its attention on the part it knows is correct. A simple neural network like ours provided with only coordinates cannot learn to distinguish well when species overlap, and will end up learning to predict the more common species in areas of great diversity.

The reason the decision boundary here looks worse for the Fourier encoding is simply because the network achieves better performance by ignoring the right half of this gaussian in favour of another class, and with the better representation of the Fourier encoding it realises this.

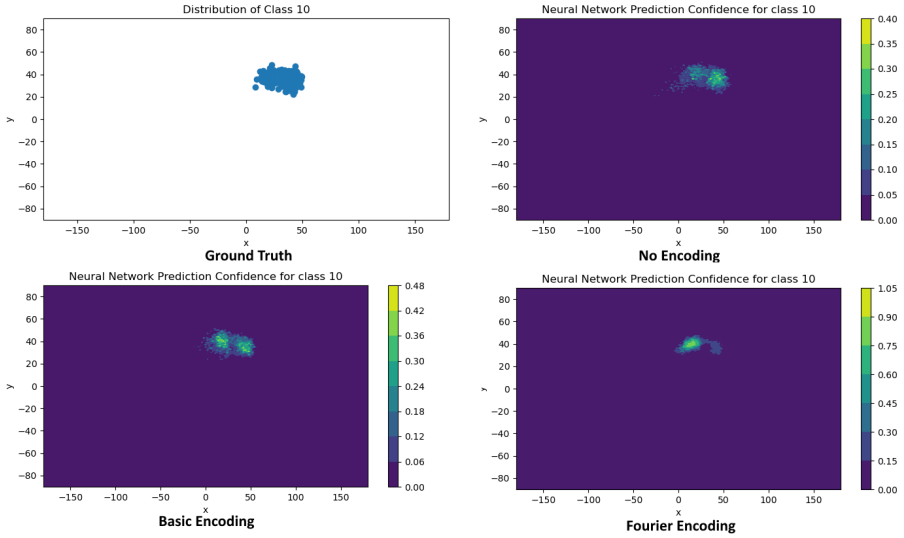


Figure 5.2: A comparison of the effects of input encodings on decision boundaries, tested on 100 classes.

This behaviour is likely the reason performance degrades so harshly as number of classes increases, as more classes leads to more areas tightly packed with many species. In this situation, one species with the most sightings will end up dominating the contested area. The most contested areas will be the clusters that most of the synthetic data is designed to be gathered around. We would expect this to continue in the iNat data since it represents a real-world scenario - there are many places across the world with a diverse amount of species in total, but a few species with vastly greater numbers. The equal classes version of the synthetic data however might not suffer as greatly from this, as we remove the limitation of the clusters - a greater performance there would provide further evidence for this theory.

5.4 iNaturalist Data Evaluation

Inspecting the iNat data like in Figure 5.3, we can see how it differs from the synthetic data - many more data points are much more spread out - in that example mainly over Russia. This gives the network very little information to use in that area, and thus it will not predict that class as present if there is another class nearby with even a few more data points, and we hypothesise this will give rise to class bias in these sparser areas. To evaluate the effects of class bias in our predictions, we use recall score as defined below, where TP are true positives and FN are false negatives:

$$\frac{TP}{(TP+FN)} \tag{5.5}$$

By calculating the recall for each class and sorting them in order, we can observe how biased the model is towards predicting certain classes. A completely fair model should produce a straight horizontal line, whilst a model that only ever picks one class should produce a straight vertical line.

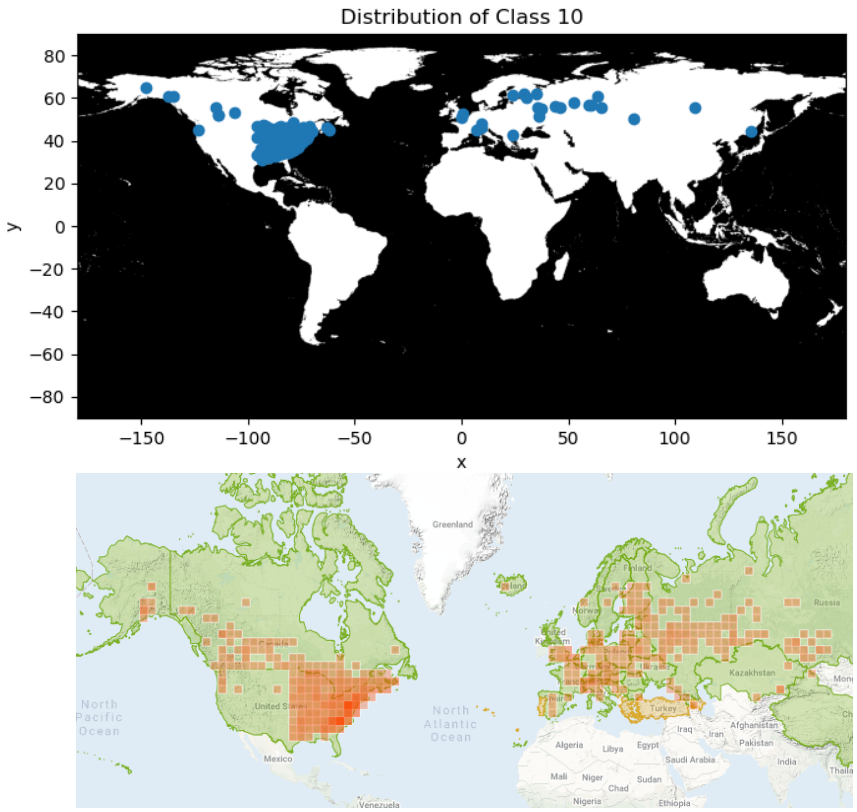


Figure 5.3: Class 10 in the dataset (top) corresponds to the Marbled Orbweaver, whose full distribution is mapped (bottom) on the iNaturalist website [4].

The results in Figure 5.4 (next page) provide a convincing display of class bias within the model. For 100 classes, over half are completely ignored and never predicted even once, achieving a recall of 0. Even worse, for 500 classes, roughly 400 are in a similar situation. This class bias is likely one of the major reasons for the degraded performance, which could also affect the performance of our model's features to the transfer learning task we wish to perform with them. It is also potentially the reason why the shape of decision boundaries for a model that achieves a better score can seem odd, as it is sacrificing some of that decision boundary to another, more prominent class, since it has learned that will achieve better results. This sharp recall decrease is not seen in the synthetic experiments (see Figure 5.5), so must involve a trait unique to iNat - we believe this to be the additional data sparsity.

There are many areas across the map where many different species are present - it would be exceedingly rare in fact for there to be a part of the world with only one species present. When we try to perform species distribution modelling only using (lat, long) values, we will inevitably run into a situation where the model has near identical inputs with two different output classes, due to this fact. Our results make it clear that in order to achieve greater performance, another variable would need to be considered. There are many choices for this contained within the iNat data. Including the date of the sighting would greatly benefit the classifier, as now it could distinguish two identical points in space over the temporal axis. There are also other properties like

the photographer in question, or the image itself that we could consider. Adding this additional variable would likely greatly reduce the class bias, as the model would be able to distinguish between the classes it previously couldn't and would not learn to ignore the less prominent classes as much.

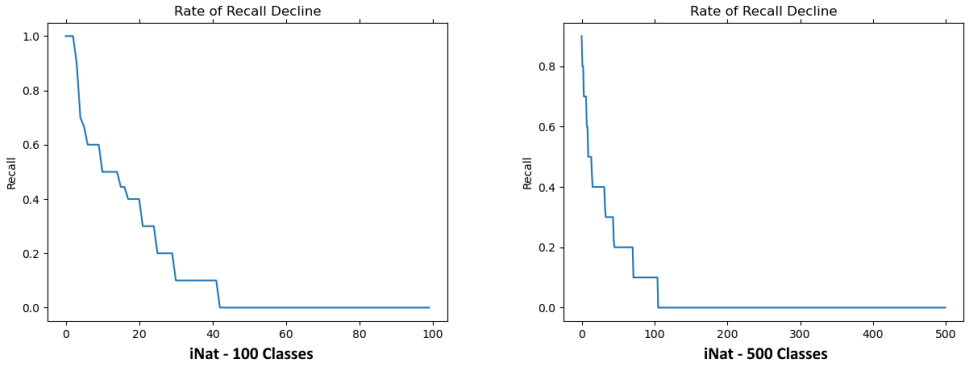


Figure 5.4: Decline in recall over the full iNat dataset, showing class bias in the model.

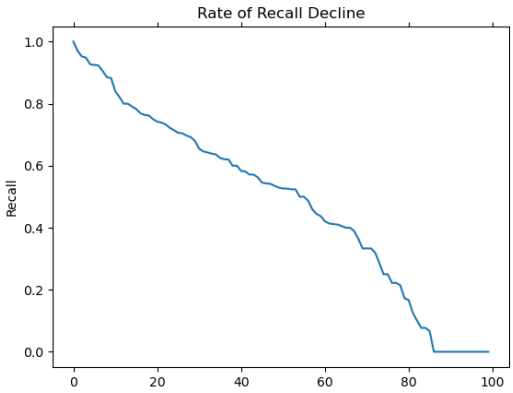


Figure 5.5: The equivalent recall graph for a synthetic data experiment using the same settings, with 100 classes. Note the much smoother decrease instead of a sharp drop.

5.5 Evaluating Equal Classes

For our purposes, we define 'equal' classes to mean a dataset where every class has the same amount of data points. For this purpose both the synthetic and iNat datasets can be initialised in their 'mini' forms. For the synthetic data, this means removing restrictions on where a Guassian is placed, and generating exactly 50 data points for each species. For the iNat data, their 'mini' dataset comes pre-split into 10,000 classes with 50 data points each, though some are missing, and we take the first N for our experiments. Such a dataset could reduce species bias in the classifier, however it may also worsen the issue of class sparsity. To investigate the effects of equal classes we train on both mini datasets to identify any emerging behaviour that differs from their 'full' forms.

Dataset Size	No Encoding	Basic Encoding	Fourier Encoding
100	0.6121	0.7468	0.8954
500	0.3512	0.4218	0.5915
1000	0.2694	0.3474	0.4604
2000	0.1851	0.2346	0.313

Table 5.3: Average precision of the various input encodings for equal synthetic classes.

For the synthetic data, performance is greatly boosted - however, this is likely not due to the equal classes and instead because we have removed the cluster restriction. This leads to the species being more evenly spread out, leading to less points of contention where the classifier has to pick between multiple likely species. With the full dataset, the classifier could achieve better performance because some species occurred far more often than others and so it could learn to be biased towards these common species.

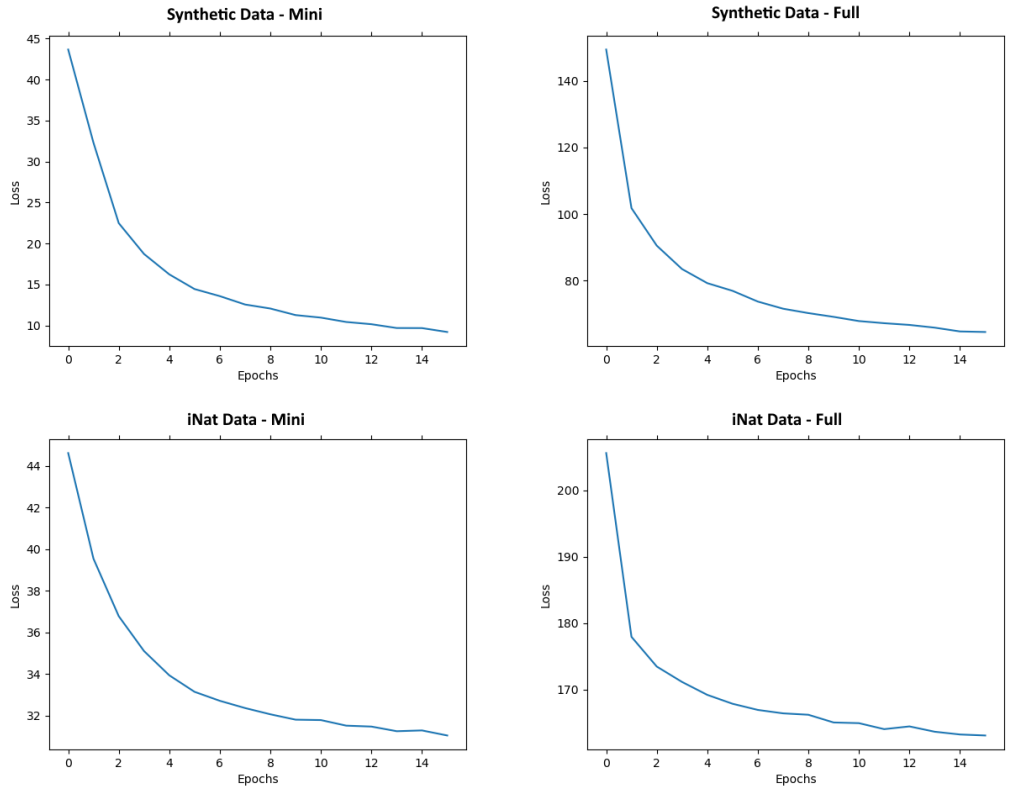


Figure 5.6: A comparison of losses during training over synthetic and iNat data, as well as their 'mini' equivalents. All were trained using the same settings (basic encoding, 100 classes).

Investigating the losses for the mini datasets in comparison to their full forms, we can see that the mini datasets are considerably easier for the classifier - their initial losses are much lower than the full datasets. The iNat mini can be seen as much more difficult however, since the model only manages to decrease it by roughly 13, whilst the synthetic data decreases that much in only 1-2 epochs. The fact it is substantially easier to train on reinforces the theory that this is because of the removed cluster restriction. If this

is the case, it would imply that the harder part of SDM for a neural model is knowing how to deal with highly populated and diverse areas as well as sparsely populated areas, which makes sense. With equal classes there is much more data sparsity and so the model has a much higher level of uncertainty in its predictions, leading to worse results. We can confirm our hypothesis by investigating the recall scores of the equal classes to see if they contain less bias.

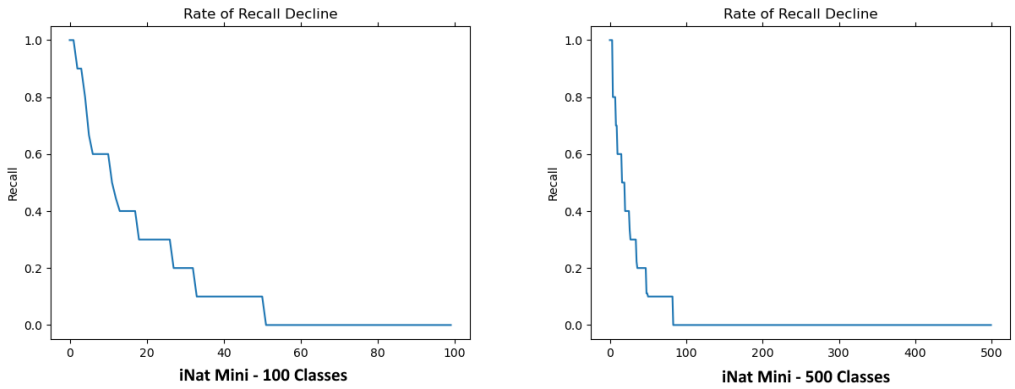


Figure 5.7: Decline in recall for the 'mini' datasets, showing class bias in the model.

These results (Figure 5.7) are interesting, as whilst for 100 classes we find that the mini dataset has more classes with greater than 0 recall score compared to its full dataset, the 500 classes case is the opposite - the mini dataset causes recall to suffer. This could be nothing to do with class imbalance, and instead to do with the increased number of classes and decreased number of data points. These lead to there being more classes that are spread out with few concentrated data points, where these classes may have had some concentrated areas in the full dataset. This would explain why the 500-class test suffers whilst the 100-class version does not, since the reduced number of classes means there are less areas with few data points from multiple classes.

5.6 Transfer Learning Evaluation

In this section we perform transfer learning as discussed in the previous chapter. We will compare the effectiveness of our pre-training approach with various input encodings to the results gathered from raw input with no pre-training, as well as a randomly initialised network (neural_random). Experiments were performed over 100, 500, 1000 and 2000 classes. To clarify, the majority of the transfer learning code was written by my supervisor, and edits were then made to this code in order to input our neural network features.

Our initial tests (Table 5.4) compare raw (lat, long) inputs with no pre-training to pre-trained features, also using no encoding (e.g., the inputs to the network were also raw lat, long).

Task	lat_long	neural_random	neural_features
ABOVE_GROUND_CARBON	0.0784	0.0303	0.0863
ELEVATION	0.4046	0.2442	0.4041
LEAF_AREA_INDEX	0.3629	0.2182	0.3715
NIGHT_TIME_LIGHTS	0.2183	0.1157	0.2317
NON_TREE_VEGITATED	0.0003	-0.0005	0.1034
NOT_VEGITATED	0.2837	0.1364	0.3085
POPULATION_DENSITY	0.2506	0.1567	0.251
SNOW_COVER	0.5387	0.0878	0.5208
SOIL_MOISTURE	0.1099	0.0689	0.1345
TREE_COVER	0.1715	0.0782	0.1912
MEAN	0.2419	0.1136	0.2603

Table 5.4: The results gathered on a baseline network with no input encoding for 100 classes, measured in R^2 as explained in Section 5.1.

This highlights the varied difficulty for each task, with tasks such as snow cover being easier for all classifiers than something like population density or above ground carbon. We can see that even without any form of encoding, the neural network approach greatly improves performance for the classification task. The hardest individual transfer learning tasks were night-time lights, followed by population density. A potential reason for this is that our iNaturalist dataset is heavily biased towards cities and other urban areas, so whilst the model has a lot of data to represent areas of relatively high population density and night-time light, the opposite is not true. This would naturally lead to issues with this particular task, compared to something like elevation which varies naturally across the world, no matter whether the area is urban or not, and therefore achieves much better performance using the SDM pre-training. Overall performance is poor, but this is what the input encodings should fix. Next we compare the effectiveness of these encodings in a similar way, but we will focus only on the mean score over all tasks.

The results in Figure 5.8 show a drastic increase in performance for the neural approaches when either encoding is applied, but the baseline model responds exceptionally well particularly to the Fourier encoding, putting it almost on-par with the pre-trained models. Still, the fully-trained SDM model does indeed provide the greatest performance over all settings we test on. This remains true for the other tested combinations of input mapping and number of classes, for which we provide details in Appendix B.

Looking at the predictions created by the pre-trained regression model, an interesting property comes to light, shown in Figure 5.10 and Figure 5.11 (next pages). Despite the SDM model performing better when there are less classes to consider, we find that including more species in our model actually improves the predictions of the fine-tuned regressor, leading to more precise boundaries that represent the ground truth data better. This is an excellent result, as it shows that it is not just the accuracy of the underlying SDM model that affects the transferred results - quantity of data is also a factor. The pre-trained model for 2000 species may not be able to predict their locations quite as well, but this extra knowledge of species diversity actually helps the fine-tuned regressor. Essentially, the more species the model knows about, the better it can use that information down the line to predict factors related to their presence or absence.

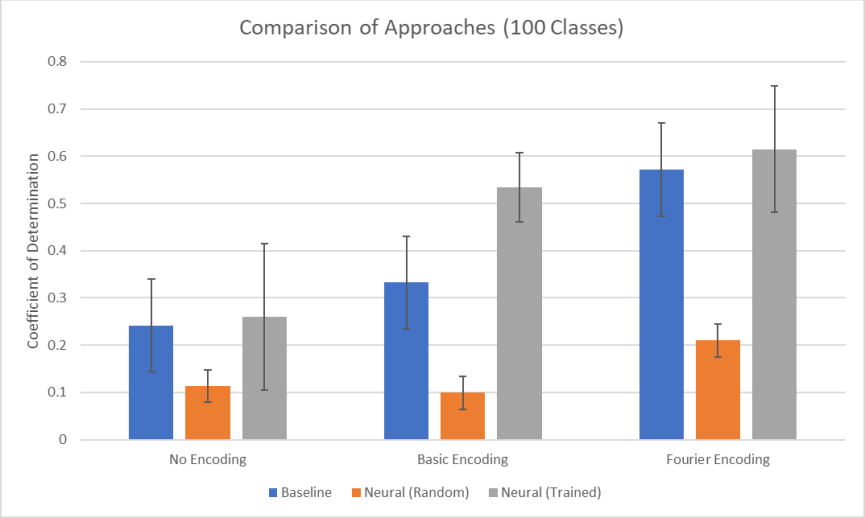


Figure 5.8: Comparison of different encodings, tested on three architectures. In the blue is the mean score without any pre-training on our SDM network. The orange corresponds to a randomly initialised network, whilst the grey shows the mean score obtained by pre-training on the network.

This result perfectly fits our intuition, and provides further evidence for the usefulness of SDM as a pre-training method.

We can even see this quantitatively by comparing Figure 5.8 to Figure 5.9 below, where we achieve a better score overall for 2000 classes. Figures B.1 and B.2 in the Appendix show the same results for 500 and 1000 classes respectively, and these perform even better than the 2000-class model. By this we can conclude that whilst increasing the number of species does help, after a point the additional complexity will harm SDM training enough for performance to begin decreasing again.

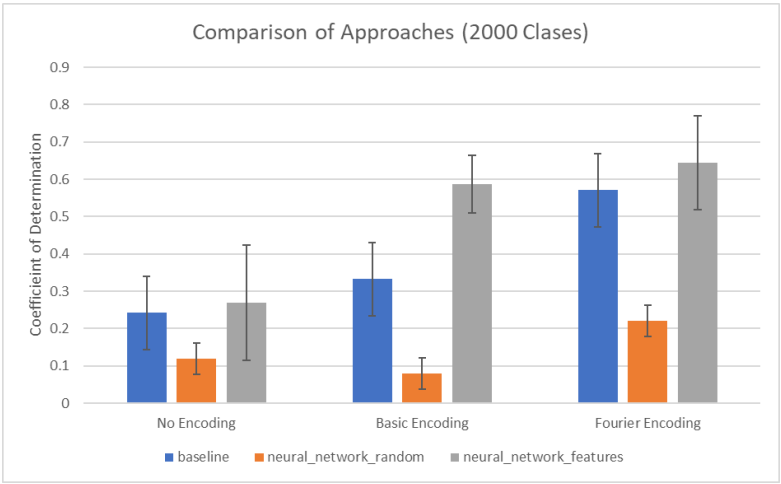
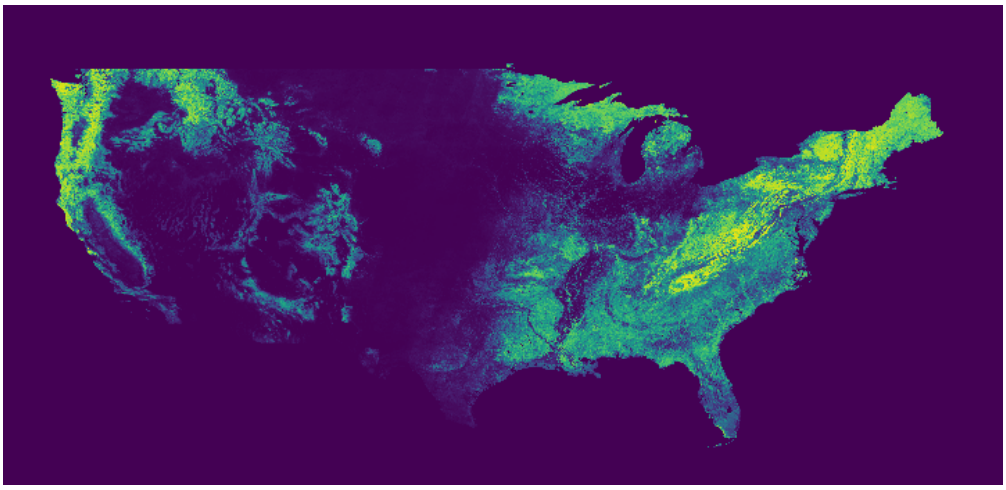
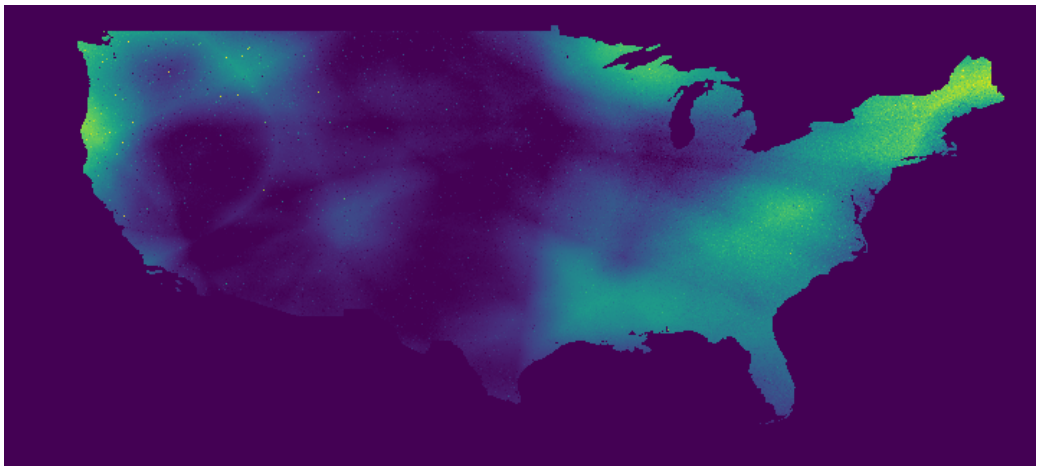


Figure 5.9: The equivalent comparison using 2000 classes - note the improved transfer learning performance.



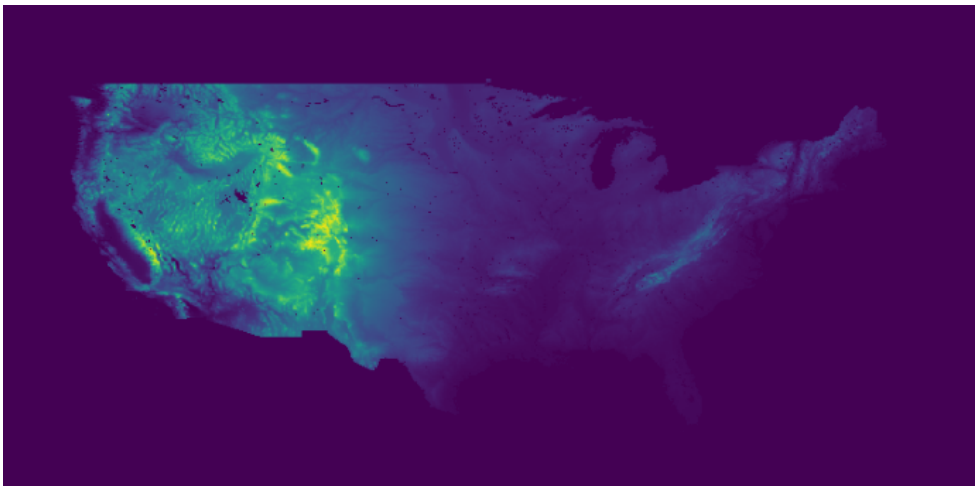
Ground Truth

100-Class Prediction



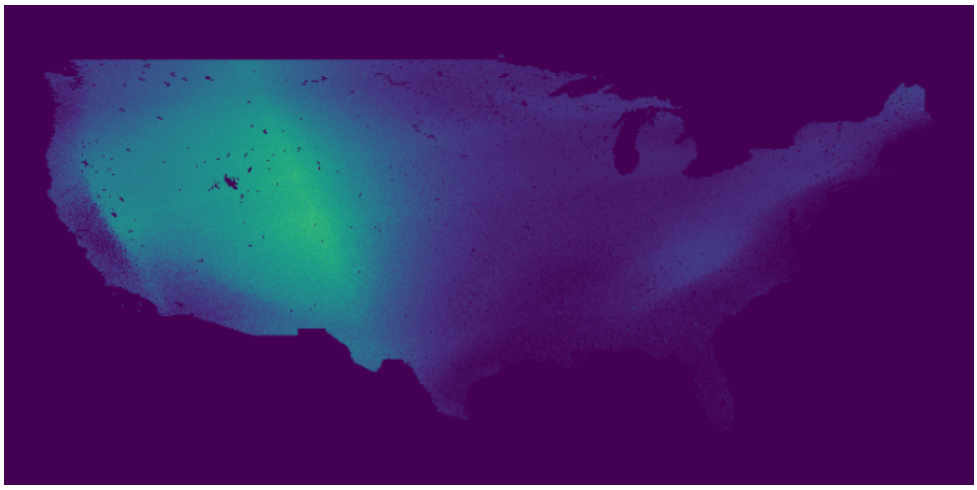
2000-Class Prediction

Figure 5.10: A comparison of the predictions when trained on differing numbers of species. The task here is Tree Cover. Note the 2000-class version is more confident ignoring the central desert areas and giving higher values to areas shown in yellow on the ground truth image. Overall the image is a lot clearer and closer to the ground truth.



Ground Truth

100-Class Prediction



2000-Class Prediction

Figure 5.11: A comparison of the predictions when trained on differing numbers of species. The task here is Altitude. Note the 2000-class version is more confident in the main areas of high altitude, and even highlights the smaller peaks to the east, whilst also giving low confidence for specific areas of low altitude like the Great Valley in California.

Chapter 6

Conclusion

6.1 Summary of Results

We aimed to answer two main research questions in this project. Firstly, can representations learned from joint species distribution models be used for transfer learning? Secondly, what is the impact of how we encode the model input on the quality of learned features to the transfer learning tasks? The first question can be answered with a clear yes - we have shown through experimental results that not only can SDM features be used as a basis for many other environmental prediction tasks, but they unilaterally improve performance over the baseline, non-pretrained approach. Performance over the set of tasks varied drastically, and particularly tasks which required a variety of data from both inside and outside of urban areas such as population density had poorer performance than universal properties due to the city-biased nature of the iNaturalist dataset. We also discovered that the number of classes has a positive effect on fine-tuning performance, even if the SDM accuracy is lowered, up until a point where the classifier suffers too much. Our results provide a convincing argument for the usefulness of SDM features as a pre-training method.

The majority of our difficulties stemmed from the performance of the SDM itself, and tuning the model to achieve greater performance would likely carry over to the various transfer learning tasks. We found our model suffered from class bias, but also came to the conclusion that a simple (lat, long) input is not enough to differentiate many classes on its own, regardless of encoding. We would require additional input information such as the date of a sighting or the actual image of the species in order to achieve greater performance.

The answer to the second question has also repeatedly shown itself in our experimental results - the choice of input encoding is absolutely vital to the quality of SDM features. A network trained using a Fourier encoding compared to one with no encoding at all consistently fits the data significantly better. Our results build a strong argument for the usefulness of Fourier encoding, not just in this particular domain, but in any application of neural networks that takes map coordinates as input. It universally outperforms the basic encoding, which in turn outperforms a network with no encoding.

In summary:

- Whilst the results gathered from our SDM itself are weak, its strength as a pre-training method cannot be denied. We can be confident that a more accurate SDM model with a more detailed representation and more species would achieve even greater performance on the transfer learning tasks and greatly outperform the baseline, non-pretrained approach.
- We have shown the importance of input encodings extensively, with particular emphasis on the Fourier encoding's excellent performance when encoding map data.

6.2 Future Work

6.2.1 SDM Improvement

Performing similar experiments using a more complex SDM network could yield interesting results and much higher performance. Our intuition that a network trained to model species distributions can be fine-tuned to other related tasks is heavily supported by the experiments performed in this paper, therefore it stands to reason that the better we are able to model species distribution, the better transfer results we could achieve. At some point improvement to the SDM network may only provide negligible gains, but pursuing better performance than our basic network is a promising area for further research. A better SDM network may also allow us to use more classes, which has previously been shown to benefit transfer learning performance.

6.2.2 Dataset Improvement

A major limitation in this project stems from the data itself, and the inherent bias within. As described when defining the iNaturalist dataset, there is a heavy bias towards urban and easily human-accessible areas that was subsequently proven to affect classification and transfer accuracy. This is an issue we have to address, as it is a natural feature of citizen science datasets like iNaturalist. Analysing the effects of bias correction methods, like the one described in [9] could be another promising direction for future research. If we could reduce the effects of bias in the dataset, it would have a heavy knock-on effect on results for the classifier.

6.2.3 Encoding Improvement

One major takeaway from the experiments is that input encoding is vital - it can drastically affect performance of the classifier, and therefore the transfer learning results. If we can improve the encoding, the whole model benefits, and this could be another avenue for future research. The Fourier encoding performed exceptionally well, but it would be of interest to test other forms of input encoding as well. For example, we could take into account the 3D nature of the globe and the distortion that occurs when stretched it to a flat map, and perform experiments on such a spherical encoding.

6.2.4 Summary

In conclusion, there are many paths for future research to take, focused on improving the individual parts of the model. Any one of these could lead to much greater performance, better features, and higher accuracy after pre-training. Species Distribution Modelling as a pre-training method has yet to be fully explored, but we have shown in this paper that it offers great potential.

Bibliography

[1] European robin (erithacus rubecula). https://www.inaturalist.org/taxa/13094-Erithacus-rubecula.

[2] Google earth engine. https://earthengine.google.com.

[3] inaturalist competition dataset 2021. https://github.com/visipedia/inat_comp/tree/master/2021.

[4] iNaturalist Website. https://www.inaturalist.org.

[5] Morgane Barbet-Massin, Frédéric Jiguet, Cécile Hélène Albert, and Wilfried Thuiller. Selecting pseudo-absences for species distribution models: how, where and how many? *Methods in ecology and evolution*, 3(2):327–338, 2012.

[6] Donald J Benkendorf and Charles P Hawkins. Effects of sample size and network depth on a deep learning approach to species distribution modeling. *Ecological Informatics*, 60:101137, 2020.

[7] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018, 2014.

[8] Daniel Chapman, Oliver L Pescott, Helen E Roy, and Rob Tanner. Improving species distribution models for invasive non-native species with biologically informed pseudo-absence selection. *Journal of Biogeography*, 46(5):1029–1040, 2019.

[9] Di Chen and Carla P Gomes. Bias reduction via end-to-end shift learning: Application to citizen science. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 493–500, 2019.

[10] Di Chen, Yexiang Xue, Shuo Chen, Daniel Fink, and Carla Gomes. Deep multi-species embedding. *arXiv preprint arXiv:1609.09353*, 2016.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Grace J Di Cecco, Vijay Barve, Michael W Belitz, Brian J Stucky, Robert P Guralnick, and Allen H Hurlbert. Observing the observers: How participants

contribute data to inaturalist and implications for biodiversity science. *BioScience*, 2021.

[13] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[14] William Fithian, Jane Elith, Trevor Hastie, and David A Keith. Bias correction in species distribution models: pooling survey and collection data for multiple species. *Methods in ecology and evolution*, 6(4):424–438, 2015.

[15] Aitor Gastón and Juan I Garcia-Vinas. Modelling species distributions with penalised logistic regressions: A comparison with maximum entropy models. *Ecological modelling*, 222(13):2037–2041, 2011.

[16] Nick Golding and Bethan V Purse. Fast and flexible bayesian species distribution modelling using gaussian processes. *Methods in Ecology and Evolution*, 7(5):598–608, 2016.

[17] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[18] Maialen Iturbide, Joaquín Bedia, Sixto Herrera, Oscar del Hierro, Miriam Pinto, and Jose Manuel Gutiérrez. A framework for species distribution modelling with improved pseudo-absence generation. *Ecological Modelling*, 312:166–174, 2015.

[19] Oisin Mac Aodha, Elijah Cole, and Pietro Perona. Presence-only geographical priors for fine-grained image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9596–9606, 2019.

[20] Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018.

[21] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[22] Duccio Rocchini, Joaquín Hortal, Szabolcs Lengyel, Jorge M Lobo, Alberto Jimenez-Valverde, Carlo Ricotta, Giovanni Bacaro, and Alessandro Chiarucci. Accounting for uncertainty when mapping species distributions: the need for maps of ignorance. *Progress in Physical Geography*, 35(2):211–226, 2011.

[23] Esther Rolf, Jonathan Proctor, Tamma Carleton, Ian Bolliger, Vaishaal Shankar, Miyabi Ishihara, Benjamin Recht, and Solomon Hsiang. A generalizable and accessible approach to machine learning with global satellite imagery. *Nature communications*, 12(1):1–11, 2021.

[24] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[25] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[26] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[27] Luming Tang, Yexiang Xue, Di Chen, and Carla Gomes. Multi-entity dependence learning with rich context via conditional variational auto-encoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[28] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[29] Peggy PW Yen, Falk Huettmann, and Fred Cooke. A large-scale model for the at-sea distribution and abundance of marbled murrelets (brachyramphus marmoratus) during the breeding season in coastal british columbia, canada. *Ecological Modelling*, 171(4):395–413, 2004.

[30] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In *International conference on machine learning*, pages 5085–5094. PMLR, 2018.

[31] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

Appendix A

iNaturalist 2021 Dataset

Super Category	Species Count	Train Images	Train Mini Images	Val Images
Plants	4,271	1,148,702	213,550	42,710
Insects	2,526	663,682	126,300	25,260
Birds	1,486	414,847	74,300	14,860
Fungi	341	90,048	17,050	3,410
Reptiles	313	86,830	15,650	3,130
Mammals	246	68,917	12,300	2,460
Ray-finned Fishes	183	45,166	9,150	1,830
Amphibians	170	46,252	8,500	1,700
Mollusks	169	44,670	8,450	1,690
Arachnids	153	40,687	7,650	1,530
Animalia	142	37,042	7,100	1,420
Total	10,000	2,686,843	500,000	100,000

Table A.1: The above data comes directly from the GitHub page for the 2021 iNaturalist competition dataset [3] and describes the data we use in the project. The test images column has been omitted, as we do not use them. With the exception of missing data, the image count represents the count of metadata (lat, long) that we are interested in.

Appendix B

Additional Experimental Results

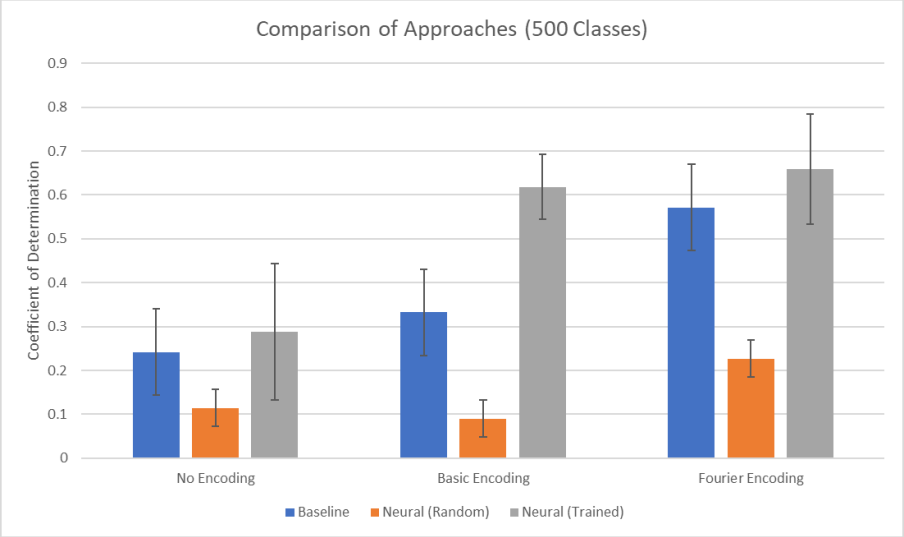


Figure B.1: Comparison of different encodings, tested on three architectures. In the blue is the mean score without any pre-training on our SDM network. The orange corresponds to a randomly initialised network, whilst the grey shows the mean score obtained by pre-training on the network.

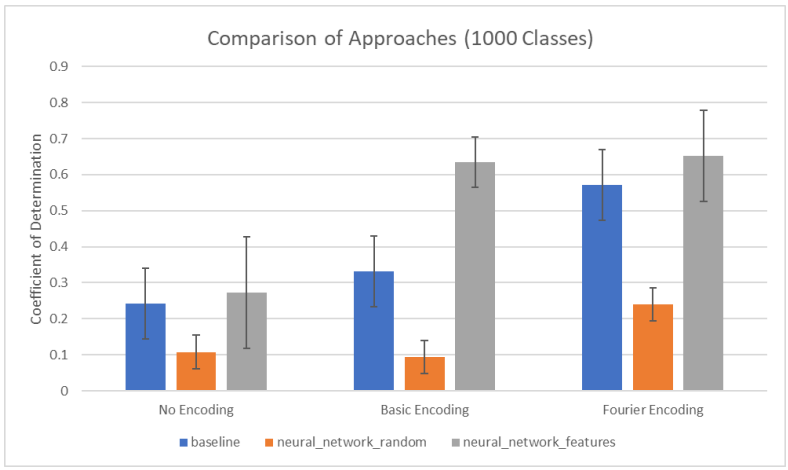


Figure B.2: Comparison of encodings for 1000 classes, with the same format as the other figures.